A method to guide assurance for Autonomous Software and Operations

Leila Meshkat

Jet Propulsion Laboratory

California Institute of Technology

Initiative Overview

Description/Goals

- Develop methods and associated tools for making informed decisions for assurance of autonomous software and operations using a modeling approach.
  - Clearly define autonomous software and operations
  - Determine the behaviors that are automated
  - Determine how these behaviors can contribute to system unreliability
  - Determine the questions that need to be addressed in order to manage the system unreliability caused by these behaviors.
  - Develop guidelines for assurance of autonomous software.

Value to NASA

- Currently there are no standard methods for conducting assurance on autonomous software.
- The products of this task will provide the SWA community with clear guidelines on the key risk elements associated with autonomous software and insight regarding areas in the system worthy of further analysis/investigation. It will also provide a means for continuous assurance of the software and commands during the lifecycle of the spacecraft.

FY18 Advancements

- Collected and synthesized relevant literature
- Report
- Developed hybrid modeling framework (Decision Trees/ Bayesian Belief Networks).
- Focus on the On-Board Planner for M2020
- Developed a suggested architectural module for "Reliability Analysis" of Plan .
- Developed Preliminary SA guidelines and BBN models

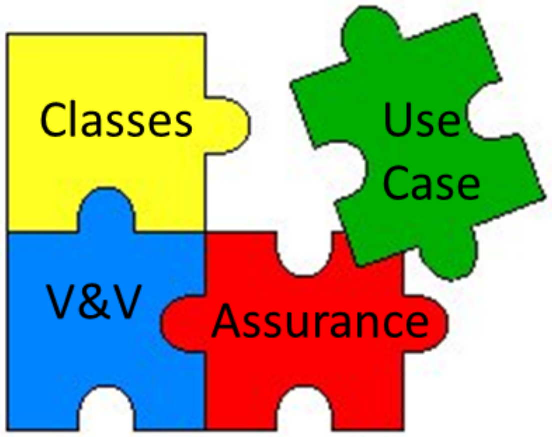
Plans (remainder of FY18)

- Iterate with M2020 team and work towards infusion
- Create journal quality paper from report
- Refine SA guidelines
- Refine models

A Method to guide Assurance for Autonomous Software and Operations

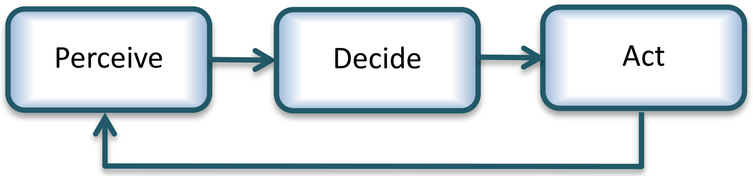
Outline

- Classes and Levels of Autonomy
  - Fault Protection
  - Command Execution
  - Planning & Scheduling
- Use Case – On Board Planner
  - Hybrid Approach
  - Proposed Architectural Updates
- FY 18 Advancements & Plans

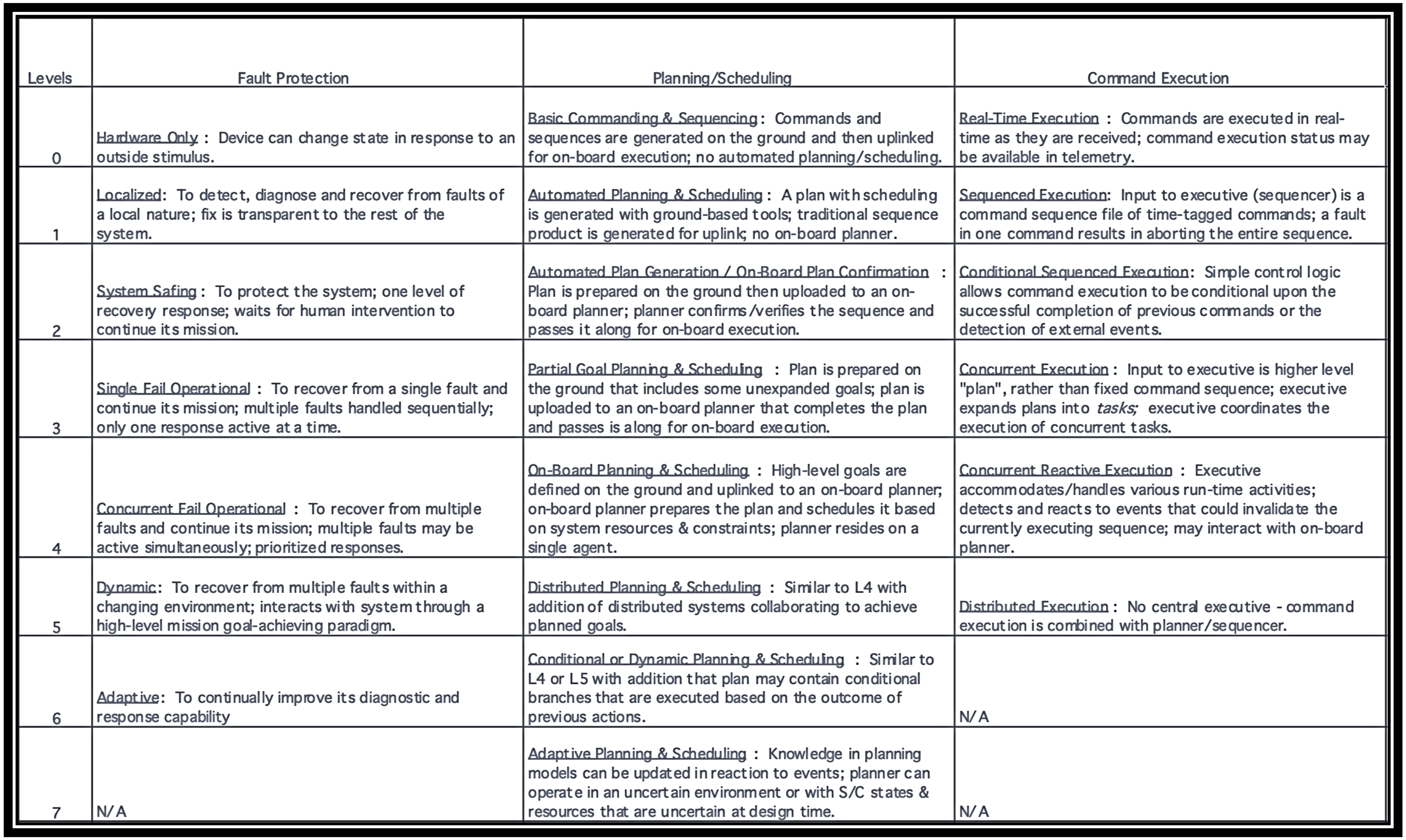


Summary of Approach

- As per the JPL Autonomy Strategic Plan, by Lorraine Fesq, et., al.,
  - Autonomy is defined, as *“making decisions and taking actions, in the presence of uncertainty, to execute the mission and respond to internal and external changes without human intervention.”*



  - Each class of spacecraft autonomy includes a set of perceptions, decisions and actions, which depend on the level of autonomy involved and correspond with uncertainties and possible unreliability for spacecraft.
    - These are summarized and provided in report for three main autonomy classes.
  - For the autonomy class, “On Board Planning” we deliberate on the methods, and architecture changes suggested for assuring the autonomy software.
- Autonomy specific guidelines are considered in addition to regular software assurance guidelines.

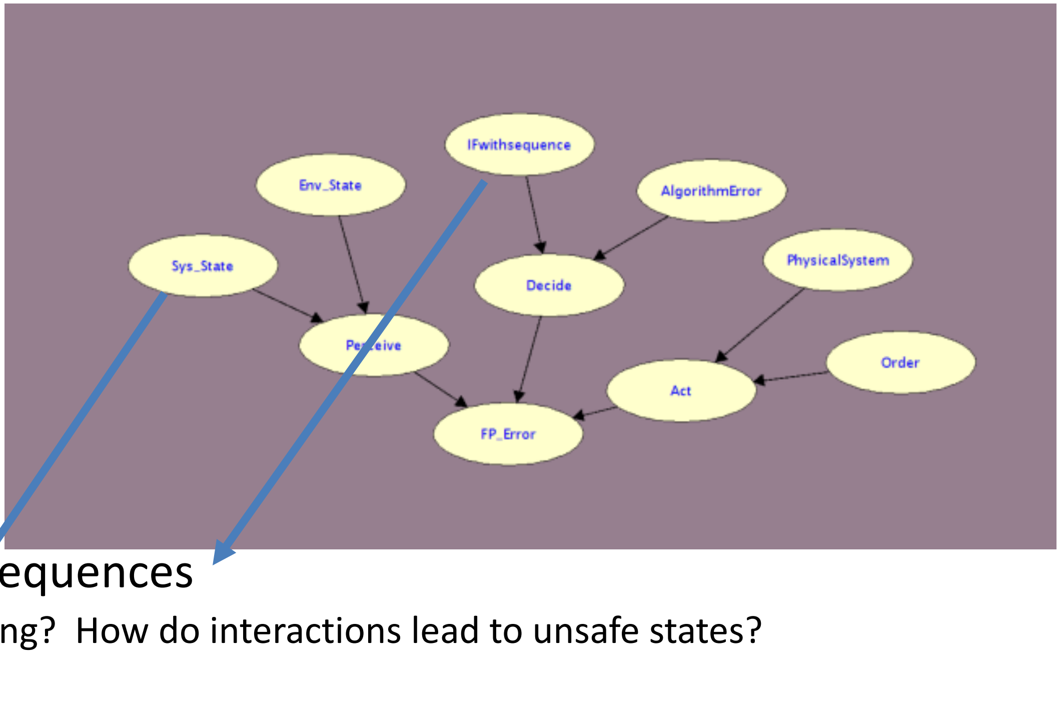


Levels	Fault Protection	Planning/Scheduling	Command Execution
0	Hardware Only : Device can change state in response to an outside stimulus.	Basic Commanding & Sequencing: Commands and sequences are generated on the ground and then uplinked for on-board execution; no automated planning/scheduling.	Real-Time Execution : Commands are executed in real-time as they are received; command execution status may be available in telemetry.
1	Localized: To detect, diagnose and recover from faults of a local nature; fix is transparent to the rest of the system.	Automated Planning & Scheduling : A plan with scheduling is generated with ground-based tools; traditional sequence product is generated for uplink; no on-board planner.	Sequenced Execution: Input to executive (sequencer) is a command sequence file of time-tagged commands; a fault in one command results in aborting the entire sequence.
2	System Safing : To protect the system; one level of recovery response; waits for human intervention to continue its mission.	Automated Plan Generation / On-Board Plan Confirmation : Plan is prepared on the ground then uploaded to an on-board planner; planner confirms /verifies the sequence and passes it along for on-board execution.	Conditional Sequenced Execution: Simple control logic allows command execution to be conditional upon the successful completion of previous commands or the detection of external events.
3	Single Fail Operational : To recover from a single fault and continue its mission; multiple faults handled sequentially; only one response active at a time.	Partial Goal Planning & Scheduling : Plan is prepared on the ground that includes some unexpanded goals; plan is uploaded to an on-board planner that completes the plan and passes is along for on-board execution.	Concurrent Execution : Input to executive is higher level "plan", rather than fixed command sequence; executive expands plans into *tasks;* executive coordinates the execution of concurrent tasks.
4	Concurrent Fail Operational : To recover from multiple faults and continue its mission; multiple faults may be active simultaneously; prioritized responses.	On-Board Planning & Scheduling : High-level goals are defined on the ground and uplinked to an on-board planner; on-board planner prepares the plan and schedules it based on system resources & constraints; planner resides on a single agent.	Concurrent Reactive Execution : Executive accommodates/handles various run-time activities; detects and reacts to events that could invalidate the currently executing sequence; may interact with on-board planner.
5	Dynamic: To recover from multiple faults within a changing environment; interacts with system through a high-level mission goal-achieving paradigm.	Distributed Planning & Scheduling : Similar to L4 with addition of distributed systems collaborating to achieve planned goals.	Distributed Execution : No central executive - command execution is combined with planner/sequencer.
6	Adaptive: To continually improve its diagnostic and response capability	Conditional or Dynamic Planning & Scheduling : Similar to L4 or L5 with addition that plan may contain conditional branches that are executed based on the outcome of previous actions.	N/A
7	N/A	Adaptive Planning & Scheduling : Knowledge in planning models can be updated in reaction to events; planner can operate in an uncertain environment or with S/C states & resources that are uncertain at design time.	N/A

Ken Clark, Paula Pingree, Garth Watney, Autonomy & Control Section 345, Jet Propulsion Laboratory, "Levels of Autonomy Technical Implementation Peer Review (TIPR)"

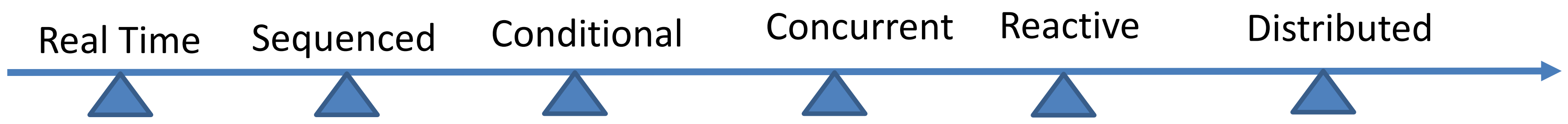
Summary of Use Case: Fault Protection

- Fault protection is defined as "the use of cooperative design of flight and ground elements to detect and respond to perceived spacecraft faults."
- Abstract Model:



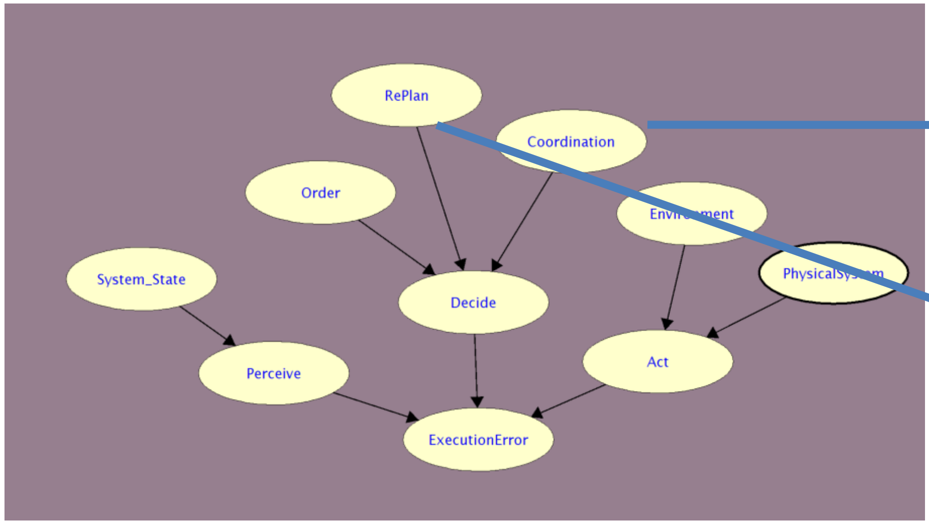
- SA Considerations:
  - Interactions with stored sequences
    - Non interacting or interacting? How do interactions lead to unsafe states?
  - System State Model
    - What is the architecture for sensing fault states?
    - What is the likelihood of error for each of the sensing mechanisms?
    - Are there interactions between the different sensing mechanisms?

Summary of Use Case: Command Execution



Increasing Levels of AI

Abstract Model:



SA Considerations:

Coordination

- What is the mechanism for coordinating between tasks?
- How do we avoid combinations that can lead to unsafe system states?

Replan

- What are the conditions that will require dynamic re-planning of sequences?
- How do we ensure correct identification of those conditions?
- How do we validate the new plan?

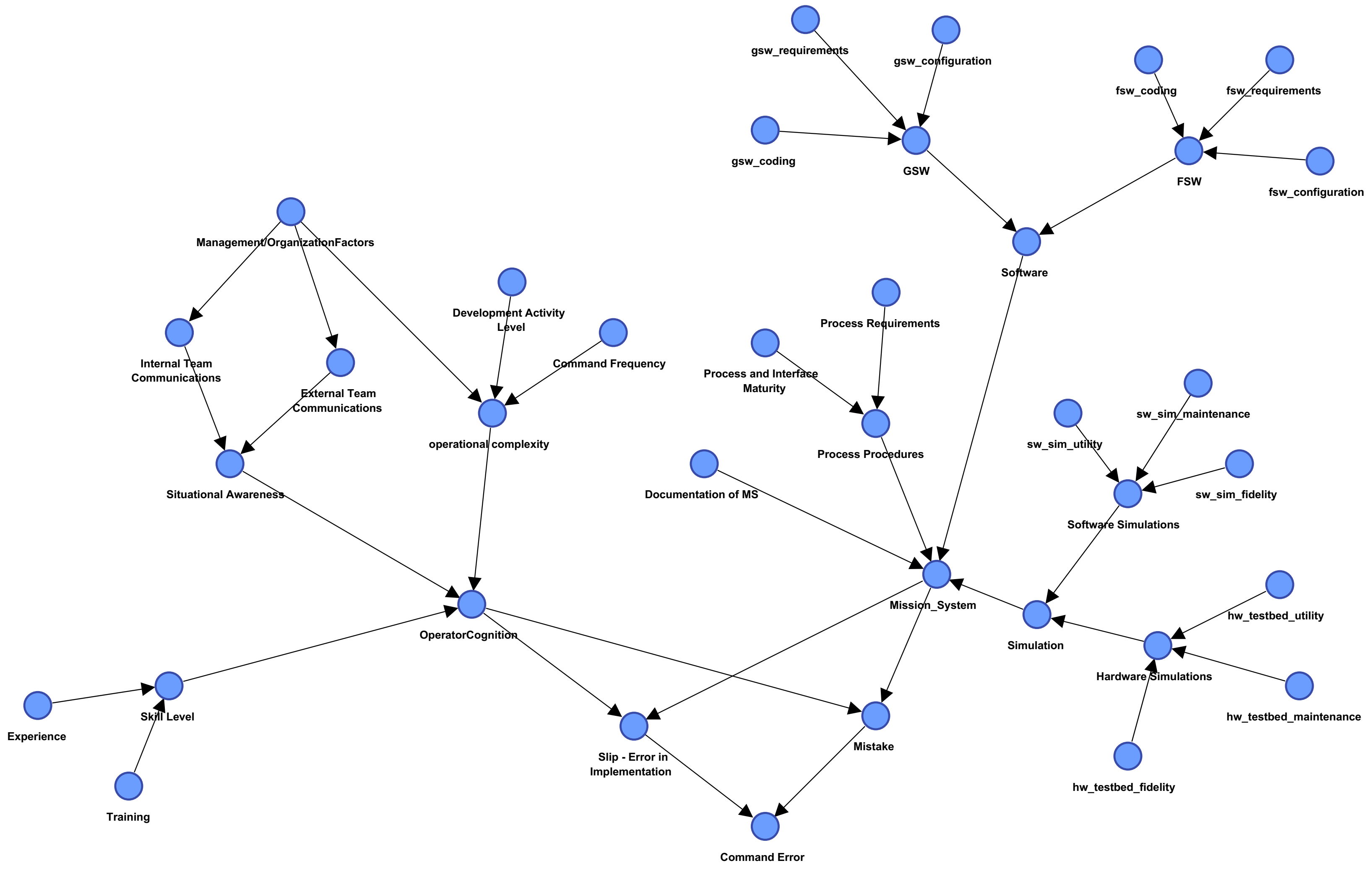
Command File Errors and Autonomy

- General Definition:
  - (a) there is an error in a command file that was sent to the spacecraft,
  - (b) an error in the approval, processing or uplinking of a command file that was sent to the spacecraft or
  - (c) an omission of a command file that should have been sent to the spacecraft

- Autonomy software does not fail due to operator error (leading cause of command file errors), however, it can fail due to the lack of knowledge of the system state or environment, or incompleteness/incorrectness of autonomy software due to its design and architecture.

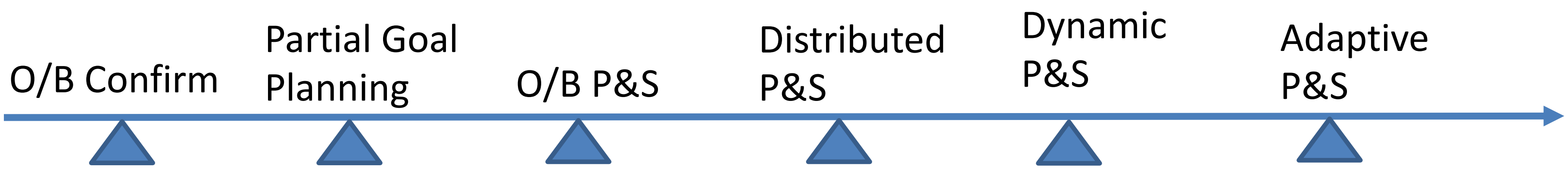
- Root Cause:
  - Without Autonomy: Operator Cognition, Mission System
  - With Autonomy: Lack of knowledge of system state or environment, incompleteness or incorrectness of autonomy software

Command File Error without Autonomy



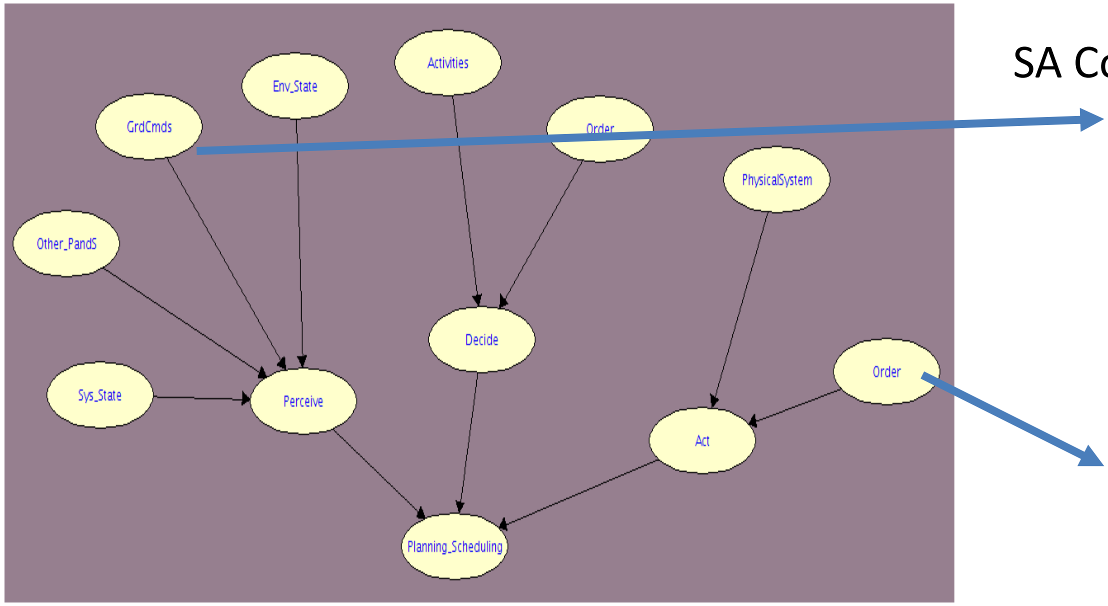
A Method to guide Assurance for Autonomous Software and Operations

Summary of Use Case: Planning & Execution



Increasing Levels of AI

Abstract Model:



SA Considerations:

Ground Commands

- What is the interplay between the ground based sequences and the on-board planner?
- What are the assumptions for each?
- What is the mechanism for their interface?
- How can it result in unsafe conditions?

Order

- What is the order in which the actions need to be taken?
- How does that manifest in the physical system?
- What are the constraints of the physical system in terms of the order of activities?
- How can they lead to unsafe states?

A Method to guide Assurance for Autonomous Software and Operations

Focus Use Case: M2020 On-Board Planner

- Failure mode identification using a hybrid approach (Functional Decomposition/ Decision & Uncertainty Analysis)

- Introducing a new "Reliability" module within the architecture to assess the reliability of a plan and use it as a trade metric when selecting the course of action.

Use Case: On-Board Planner on M2020

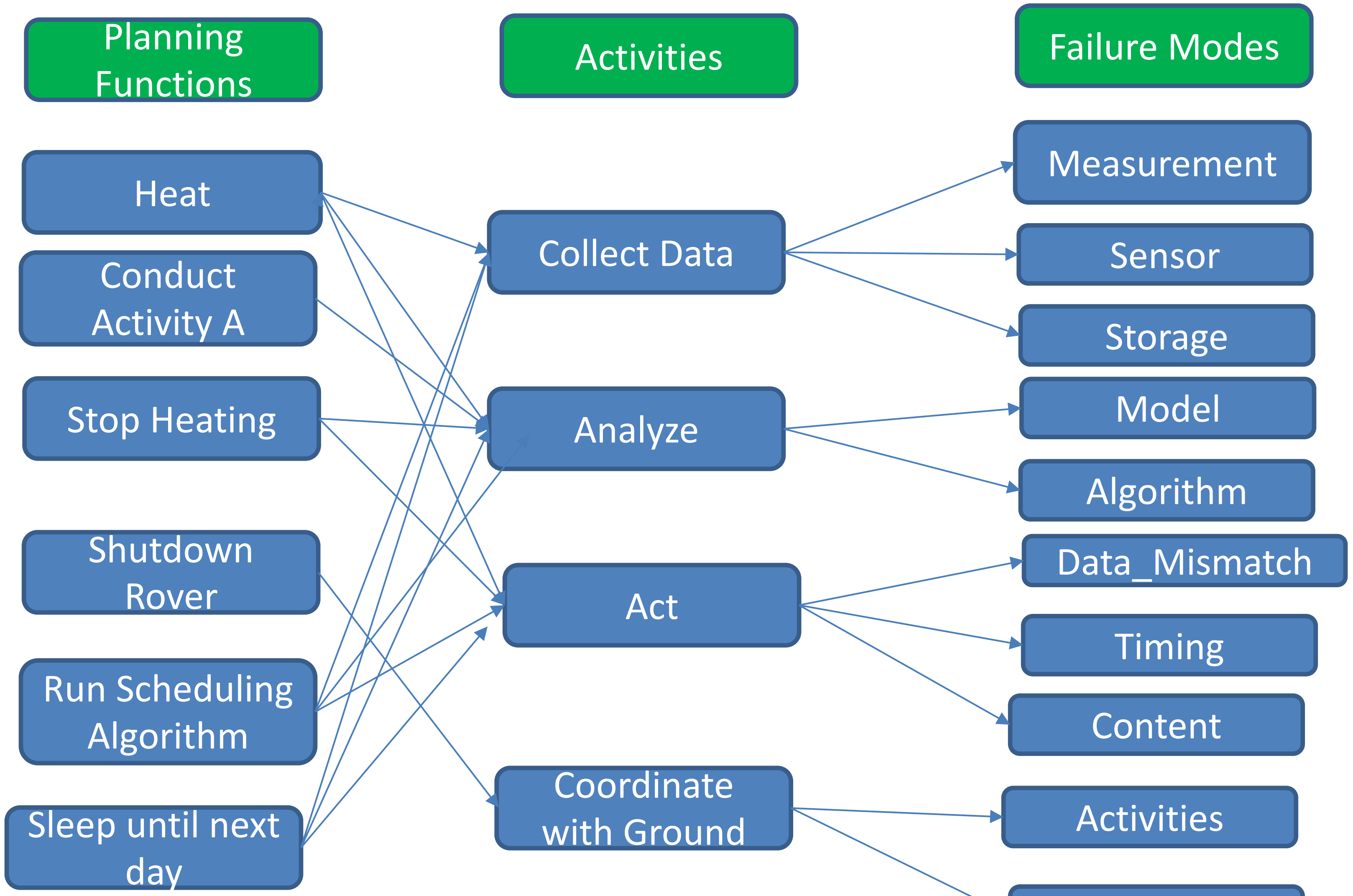
- Approach:
  - Identify Key autonomous decisions.
  - Identify possible uncertainties – outcomes
  - Build Decision Trees
  - Assess likelihood of each branch
  - Feed into BBN modeling

Use Case: Key Decisions

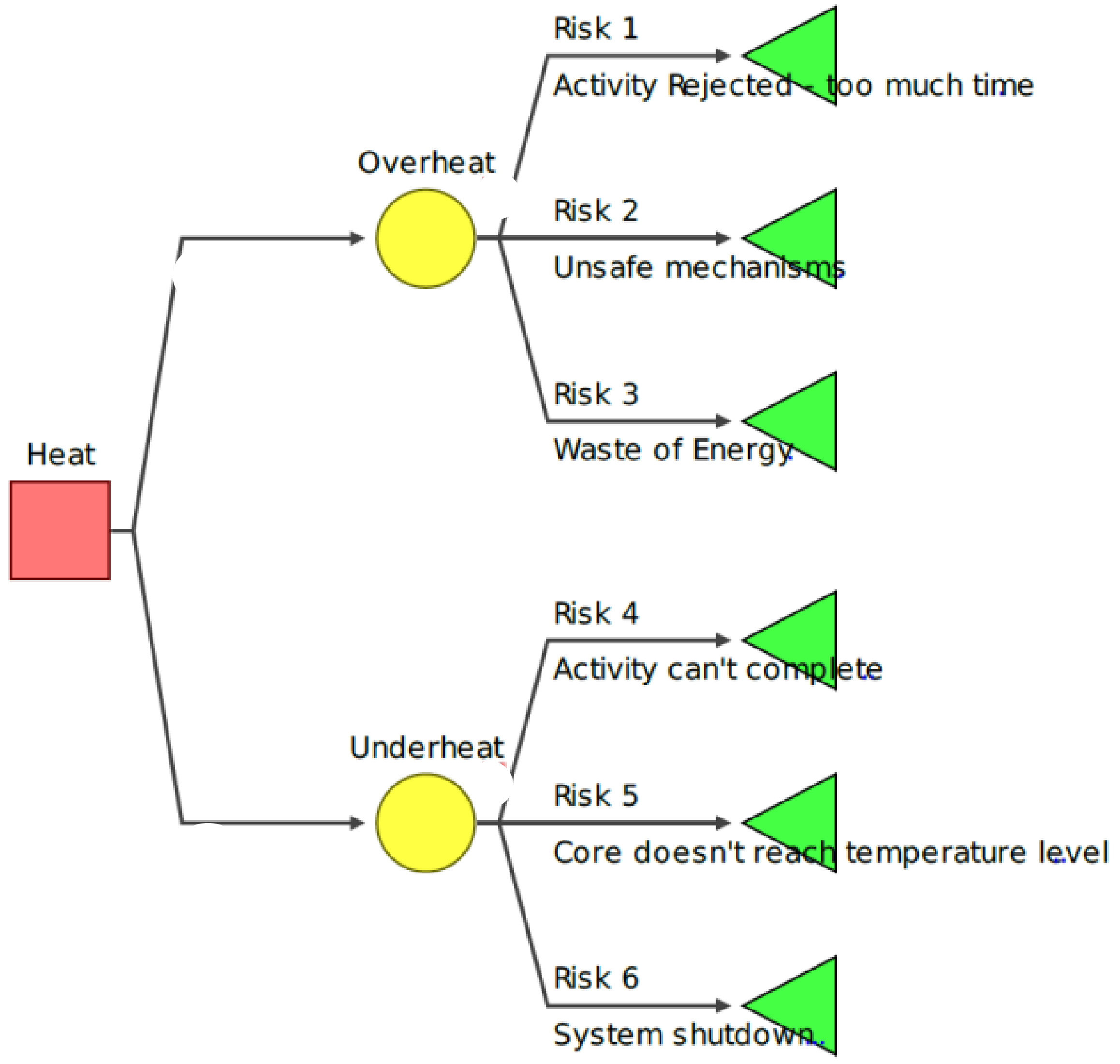
- Whether or not to schedule a maintenance or heat activity
  - The heating needs to be scheduled before the activity it's supposed to use the heat for.
    - What can go wrong?
      - You may not finish the heating in time and the activity is rejected
      - The system thinks it's heated and it is not . You could use mechanisms that are not safe to use.
      - You could be heating more than you need to and that can lead to a waste of energy.
      - If the core doesn't reach the temperature
      - If a given activity takes much longer than it needs to it won't be able to perform with pre-planned heat levels.

- For a set of activities (a1, …. an), what is the correct order?
  - Which activity goes first?
  - Which activity goes second?
  - Etc.

- Time for the vehicle to sleep or wake up.

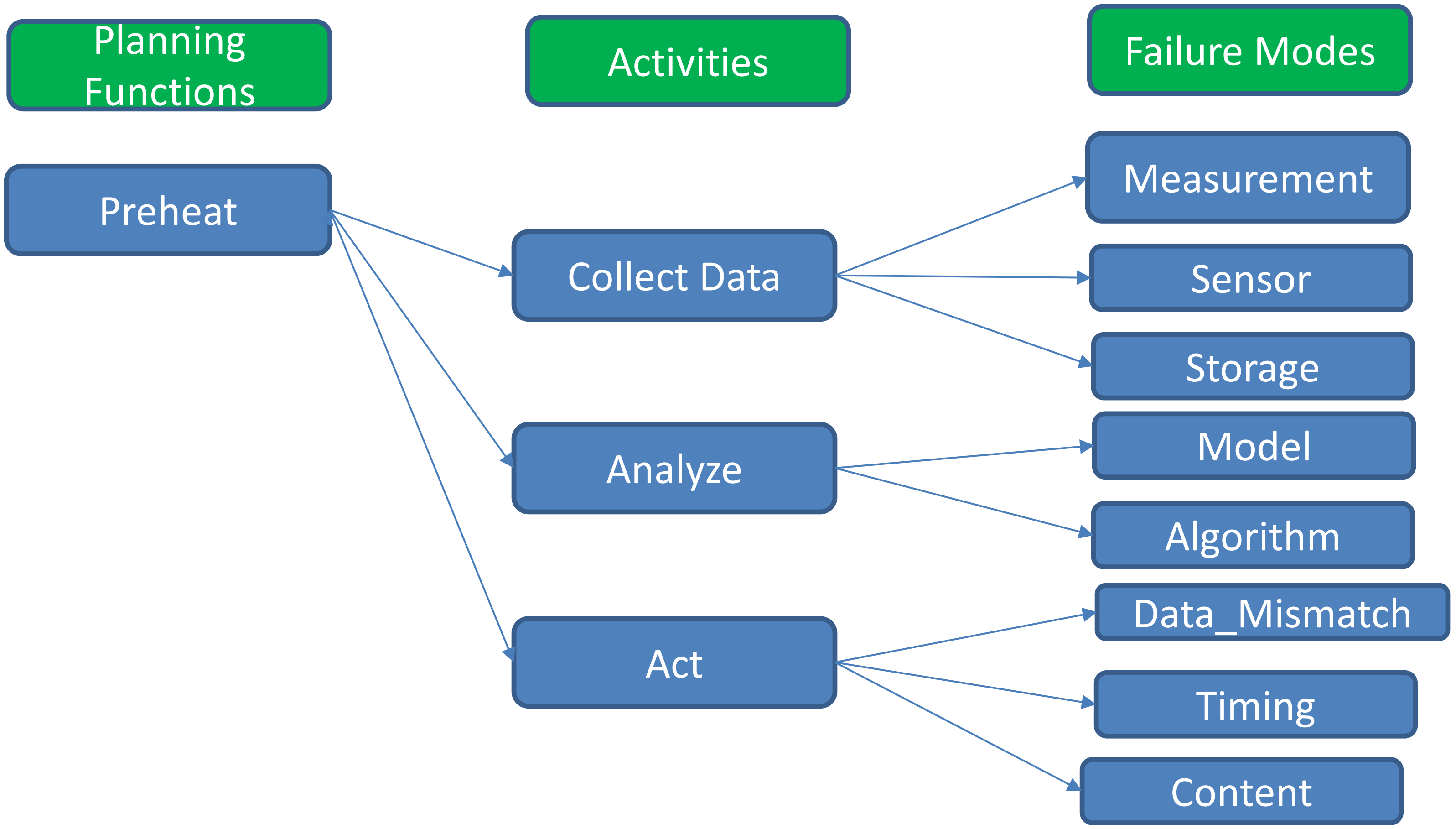

Planning Functions
Activities
Failure Modes
Heat
Conduct Activity A
Stop Heating
Shutdown Rover
Run Scheduling Algorithm
Sleep until next day
Collect Data
Analyze
Act
Coordinate with Ground
Measurement
Sensor
Storage
Model
Algorithm
Data_Mismatch
Timing
Content
Activities
Order

Sample Decision Tree at a given point in time

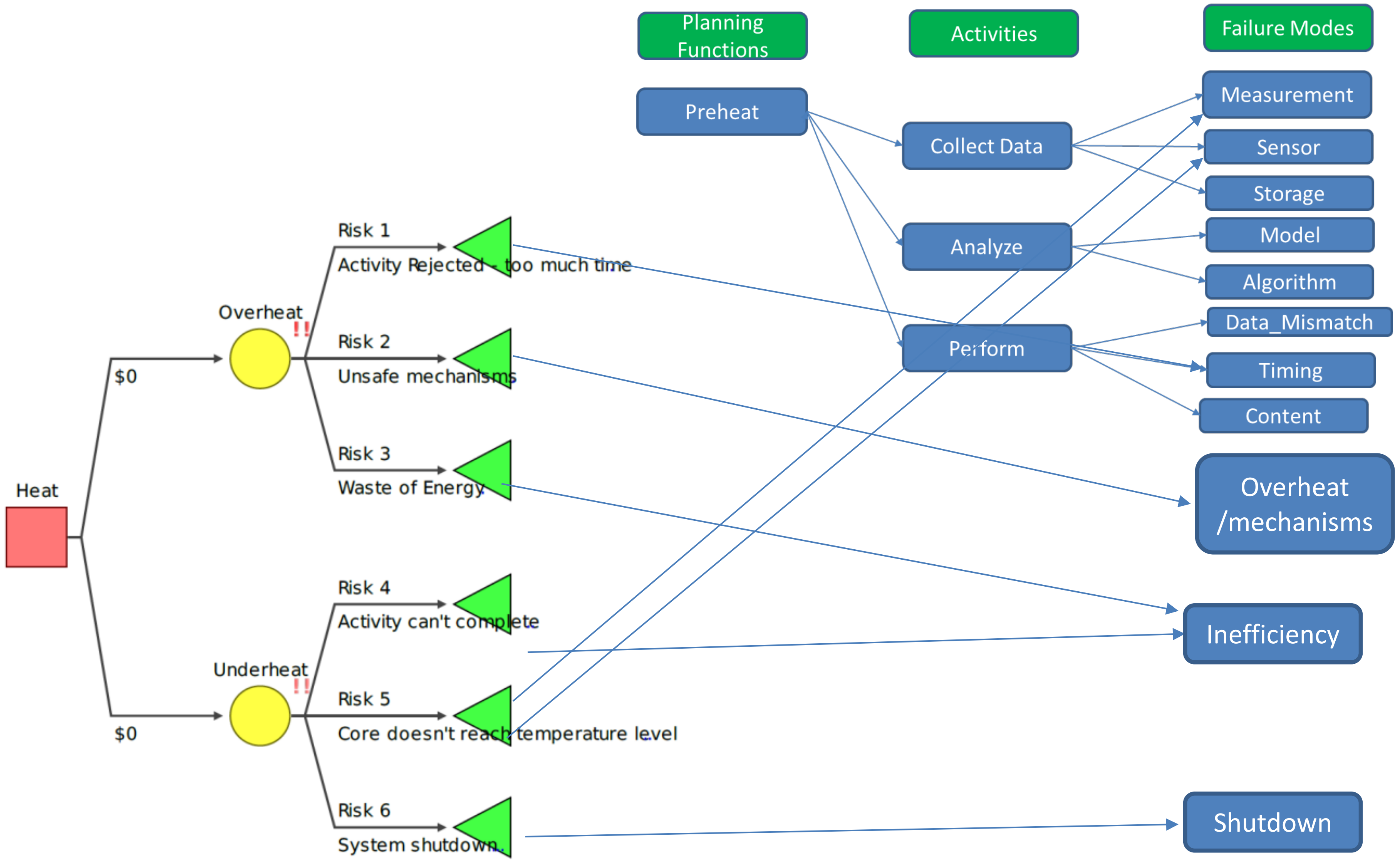


A Method to guide Assurance for Autonomous Software and Operations

Sample BBN Structure



Hybrid Approach

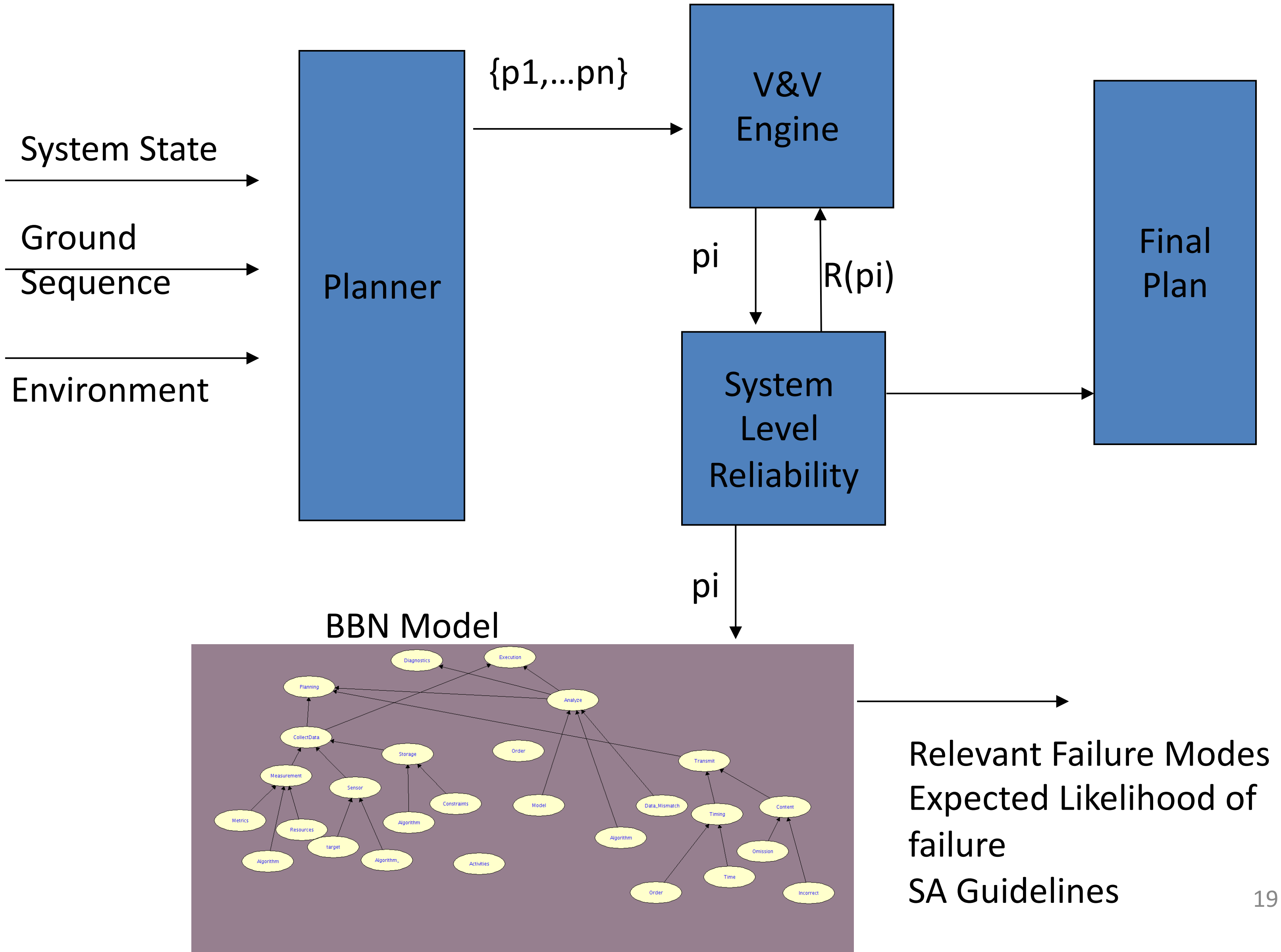


A Method to guide Assurance for Autonomous Software and Operations

V&V Versus Assurance

- Currently, there are R&TD tasks addressing the issue of V&V for the On-Board Planner
- Candidate plans are assessed for completeness and feasibility
- We propose to augment this V&V engine with a module to assess the system level reliability of proposed plan
- The goal of this module is to ensure that the plan does not lead to unsafe states for the system

Proposed System Architecture



FY18 Advancements

- Collected and synthesized relevant literature
  - Report
- Developed hybrid modeling framework (Decision Trees/ Bayesian Belief Networks).
- Focus on the On-Board Planner for M2020
- Developed a suggested architectural module for "Reliability Analysis" of Plan .
- Developed Preliminary SA guidelines and BBN models

Plans – FY18 (& beyond)

- Iterate with M2020 team and work towards infusion
- Create journal quality paper from report
- Refine SA guidelines
- Refine models